Table of Contents

[Introduction 2](#_Toc122647733)

[Exploration 2](#_Toc122647734)

[Preprocessing 3](#_Toc122647735)

[Duplicates and new features 4](#_Toc122647736)

[Strange values and outliers 4](#_Toc122647737)

[Missing values 4](#_Toc122647738)

[Data types 5](#_Toc122647739)

[Data reduction 5](#_Toc122647740)

[Feature selection 6](#_Toc122647741)

[Modelling 7](#_Toc122647742)

[Assessment 9](#_Toc122647743)

[Conclusion 9](#_Toc122647744)

[References 10](#_Toc122647745)

[Annexes (Todos) 12](#_Toc122647746)

[Self-study – data exploration 29](#_Toc122647747)

[Self-study – feature selection 31](#_Toc122647748)

[Self-study – models 32](#_Toc122647749)

# Introduction

This report focuses on creating a predictive model as the final project in the Machine Learning course. For the development of the project a hypothetical situation was given, where we had to analyse a group of patients with different sociodemographic, health, and behavioural information and predict which ones have a higher likelihood of contracting the Smith Parasite, a disease that, until now, has been contracted without apparent connection between individuals. In other words, our main concern is answering the question “Who are the people more likely to suffer from the Smith Parasite?”.

In the following pages, we present and discuss the entire process of the project development including data treatment, models created, decisions made, and major findings.

# Exploration

Data visualization is a crucial step in data analysis as it provides us with a graphical representation of our information and data. It helps us see the “big picture” and enables the viewer to start making assumptions on what will be some of the patterns in our dataset, therefore saving time as we are aware of what steps to prioritize.

We started by importing the required librariesand checking the number of elements in our dataset. In table 1 we can see all our 18 variables and a brief description. After checking a concise summary of the data frame, we can see that *Education* is the only one with missing values. We can also see there are some variables, such as *Smoking\_Habit* and *Exercise*, that are wrongly stored as strings, instead of binary values, which can later cause some adversities.

Following this, we started to inspect the main descriptive statistics for numerical and categorical variables, as shown in tables 2 and 3. The first problem we noticed was that the minimum value of the variable *Age* (1855) seemed like an outlier, since the Smith Parasite was recently discovered, so we should expect the people from our dataset to be alive in 2022. Also, looking at *High\_Cholesterol* maximum value, we could see that it is far away from the third quartile, so probably we are in the presence of an outlier (maybe an error of measurement, since the normal range of the cholesterol in the blood is <200). Finally, the *Blood\_Pressure* maximum value is considered anormal, since the medically normal range is 90-120, and it is considered elevated in the range 140-180. Another problem that we noticed in this dataset was the repetition of a Name (Mr. Gary Miller appears twice), so we should check for duplicates during the preprocessing phase.

For further analysis, we decided to use plots to identify distributions and irregularities in the pattern of our data. Then, we divided the variables into numerical, categorical, and ordinal features as seen in table 4.

By analyzing the histogram plots of the numeric features shown in figure 5, we can see some clear outliers in *Birth\_Year,* *High\_Cholesterol*, and *Physical\_Health.* Furthermore, *Height* and *Mental\_Health* are skewed to the right which indicates that their values tend to be on the higher side. On the other hand, *Blood Pressure*, *High\_Cholesterol*, and *Physical\_Health* are skewed to the left, so the individuals in our train dataset tend to have lower values in the variables.

When we use box-plots to represent the numeric variables, shown in figure 6, we easily spot outliers in *Birth\_Year*, *High\_Cholesterol*, *Blood\_Pressure*, and *Physical\_Health*, and these results emphasize the conclusions already found with the descriptive statistics. These box plots will be very useful in the outlier removal process, as sometimes the IQR method is not the most efficient.

For numerical variables, we can still analyze the pairwise relationship, as seen on figure 7. There are various possible multidimensional outliers, which are the points we can see outside the big clusters of points. Also, there seems to be a linear correlation between *Height* and *Weight*. The other correlations will later be evaluated using Spearman correlation which assesses relationships between variables, even if the relationships are not linear.

For non-metric variables, we created bar plots, to extract useful information, shown in figure 8. The variable *Name* is excluded because it won’t provide valuable interpretation. From these bar charts, we can say that most of the patients in our training dataset do not smoke more than 10 cigars daily and do not exercise (more than 30 minutes) 3 times per week or more. Additionally, most of the patients did not make any check-ups in the past 3 months. The most common region of the patients is ‘East Midlands’ and they have a complete university degree. Specifically looking at the *Region* bar chart, we can see that there is an inconsistency, since ‘London’ appears in two bars, as the code is case sensitive, so we should correct this.

Finally, we decided to create some visualizations comparing how the variables behave when there is and isn't disease. The first plot, shown in figure, provides additional information on the influence of the target on the pairwise relationships of the numerical variables. In the box plot, figure , we can see some minor differences such as the patients with this disease had more days where they felt their mental health was affecting their life and fewer days where their physical health was in a bad state, compared to patients who weren’t diagnosed. Doing the same for histograms, figure , we can see some differences in *Height*, *Weight*, *Physical\_Health* and *Mental\_Health*, which can indicate some influence of the target variable, even if not too significant, in these specific variables.

Alternatively, on the non-metric variables (figure ), we can see there is a higher contrast in the variables which are affected by the diagnosis of the disease (for example, *Checkup*, *Diabetes* and *Exercise*). It shows most of the infected patients exercise less than 3 times a week, are most likely to have more than 3 years before the last check up and have had pregnancy diabetes or borderline diabetes, which differs from the most frequent values when we consider all the patients.

# Preprocessing

We know that databases are susceptible to noisy, missing, and inconsistent data (Han, Kamber, & Pei, 2012) and that low-quality data will lead to low-quality results from our models. That is why data preprocessing is an important step in any machine learning project. It involves cleaning and formatting data to make it suitable for algorithms to use it. Knowing that splitting a dataset into training and testing sets is important because it allows us to assess the performance of the models on unseen data, we should apply some steps of data preprocessing only on the training dataset. This is crucial because we want to build a model that can generalize well to new data, rather than just memorizing the training data. To avoid overfitting, we should not include any validation and training information in our train dataset, so, for example, if we are using a measure of central tendency to input the missing values, we should calculate the measure for the training dataset, and use it to fill the missing values in all the sets (training, validation, and test).

To select the best model, we should have a validation dataset. We can use several methods, but **cross-validation** and **hold-out** are both methods used to evaluate the performance of a machine-learning model. The main difference between the two is that in cross-validation the data is split into several subsets, and the model is trained and tested using different combinations of these subsets, while the hold-out method uses a fixed split of the data for training and testing, which can result in a less accurate evaluation of the model if the split is not representative of the data. So, since we used cross-validation on our project, we just defined and tested the appropriate functions to preprocess the data and then on the modeling phase we applied all these functions to the different sets we were using.

## Duplicates and new features

There were no duplicate rows, so we assumed that the two patients with the same name (‘Mr. Gary Miller’), but different IDs and different information, was merely a coincidence. Then we created three new features (see table) that we thought might be useful to predict our target: *Age*, *Gender*, and *BMI*.

## Strange values and outliers

Looking at the bar charts (of non-metric features) from the exploration step, we solved the problem with the *Region* variable, and after that we looked on our dataset for **incoherences**, for example, if there were patients under 16 years that already had a college degree.

We tried two different methods to identify **outliers**: the IQR method and manual filtering. Using the interquartile range, an observation was an outlier if it had an absolute value 1.5 times greater than the IQR, but we would have to remove 8% of our data, which was considered too much, and therefore, not applied. For the manual approach, we defined some conditions that our observations had to satisfy to not be considered outliers. The high values in *Blood\_Pressure* and *High\_Cholesterol* found in the exploration phase are unlikely to happen, but they are still possible, so we should consider them. We created a function to delete the observations referring to patients that were born before 1900, in the training dataset.

## Missing values

As we know, missing values are usually expressed with NaN (as we found in *Education*), but it was further checked if there could be other expressions for missing values, but none were found.

First, we tried to use an adequate **measure of central tendency**, which in our case is the mode, as we are dealing with an ordinal feature. Second, we tried to use the **mode** of *Education* for all samples belonging to the **same class** as the given observation (Han, Kamber, & Pei, 2012). For example, we may replace the missing value with the mode *Education* value for patients from the same *Region* category as that of the given observation. Third, we considered **deleting** the observations with missing values, since the 12 observations with missing values represent 1.5% of the training dataset. We should keep in mind that in this way we would be losing information to train the model. We also could have deleted the feature *Education*, but "as a general rule of thumb, only features that are missing in excess of 60% of their values should be considered for complete removal” (Kelleher, Namee, & D'Arcy, 2015), which is not the case. "There are other, more complex approaches to imputation. For example, we can build a **predictive model** that estimates a replacement for a missing value based on the feature values that are present in a dataset for a given instance" (Kelleher, Namee, & D'Arcy, 2015). So, we tried to predict *Education* based on the other features, using a Gaussian Naïve Bayes model.

We should start with the simplest method, so we decided to use the imputation of the mode as the method to deal with missing values, and then change the method if needed to improve the performance of our models.

## Data types

Machine learning models require all input and output variables to be numeric (Brownlee, 2020), so we used two forms of **encoding for non-metric features**: one-hot encoder and ordinal encoder. One-hot encoding was used to encode categorical features (the ones defined above in the exploration phase) for which no ordinal relationship exists. When encoding our dataset, we noticed that a category ‘Education\_nan’ was created, so in the modelling phase, we should first solve the problem with missing values and then encode the variables. Ordinal encoding was used for the ordinal features (the categories have a naturally ordered relationship with each other), and we defined the specific order we wanted inside the OrdinalEncoder instance. The created functions can encode the non-metric features specified as input (and not only the adequate ones if we want).

Since we will use methods that are based on measures of how far apart data points are, we need to scale our data. By **scaling** our variables, we can compare different variables on equal footing, because, for these algorithms, a change of "1" in any numeric feature is given the same importance (Cook, 2021). We started by trying min-max scaling, as it will preserve the shape of the dataset (no distortion) and it is the least disruptive to the information in the original data. Then, we tried the standard scaling, but the problem with this type of scaling was that it should only be used when we know that the data distribution is normal (which is not the case). Lastly, we used robust scaling that scales the data according to the quantile range, so outliers will have less influence than in other scaling methods. After checking the results on the whole training dataset, we noticed that the range of the variables when using robust scaler was in most cases bigger than when using min-max scaling, so we decided to keep the min-max method.

## Data reduction

Dimensionality reduction is the transformation of data from a high-dimensional space into a low-dimensional space (Han, Kamber, & Pei, 2012). We used **Principal Component Analysis** to achieve this, but we ended up not using it, because there is a problem with interpretability of the principal components, as we cannot understand how the features are contributing to separate the target classes.

## Feature selection

“Relevant features lead to efficient models, not more features! Note that including a large number of features might lead to an overfitting problem” (Swamynathan, 2017), consequently continuing to add new features to the dataset results in a decrease in the predictive power of the models. For some methods, we need to split the data and deal with the preprocessing problems after (using the functions defined before).

* **Filter methods:**

Firstly, we started by checking for any univariate variables, because if a feature has a variance of 0 it does not provide any predictive capacity, so we should drop it. Then, we computed the Spearman correlation that assesses monotonic relationships and looking at the correlation (see figure), we concluded that there was no independent variable highly correlated with the target, and there were two pairs of variables highly correlated, namely: *Weight* vs. *BMI* (correlation of 0.8) and *Age* vs. *Birth\_Year* (correlation of -1).

We performed ANOVA (see table) and, looking at the p-values, we concluded that we reject the null hypothesis for all metric features at a 5% level of significance, meaning that we have statistical evidence that the means of the patients with the disease and the mean of the patients without the disease for every metric feature are different. However, we should not include *High\_Cholesterol* in the model because its p-value is close to the significance level. Looking at the p-values of Kendall's rank correlation coefficient (see table), we conclude that we reject the null hypothesis for all metric features at a 5% significance level (meaning that we have statistical evidence that all features and the target are statistically dependent), so we do not discard any metric variable. However, this test can also be used with ordinal features, so looking at the p-values we consider that *Education*, and W*ater\_Habit* and the target are statistically independent, hence they are not important predictors.

Using the chi-squared test, we concluded that we should discard the variables *Name*, *Region*, *Education*, *Smoking\_Habit* and *Water\_Habit*. Also, with common sense, we can understand that *Name* will not be a good predictor for whether a patient has the disease or not, since it is a unique identifier of the patient, so, in our further analysis, we will not use it. According to mutual information, we should discard *Water\_Habit*, *Smoking\_Habit*, and *Region*, since they have mutual information of 0 (see figure), meaning that they are independent of the target. Also, we can look the same way for *Education* (with mutual information very close to 0). *Exercise* and *Fruit\_Habit* seem the most important features regarding the mutual information coefficient. Also, *Diabetes* and *Checkup* have two categories that are considered important and one category that is not, therefore, we should keep them.

We also visualized the weight of the target in the non-metric features (see figure). For example, we can conclude that females have a higher probability to have the disease than males, so maybe *Gender* is an important variable to predict the target.

* **Wrapper methods:**

We used RFE (recursive feature elimination) combined with logistic regression and a decision tree (that assigned weights to the metric features). We created a function that gave us the ideal number of features to keep and provided RFE with this number. According to RFE, combined with logistic regression, the numerical variables we should discard are *Birth\_Year* and *Age*. When combined with a decision tree, the numerical variables that we should discard are *Weight*, *Height*, *Age*, and *BMI*.

* **Embedded methods:**

We used Lasso regression, which picked 6 variables and eliminated 3 other variables, and concluded that we should discard *Weight*, *Birth\_Year*, and *Age*, but *BMI* also seems insignificant for the definition of the target compared to the remaining predictors. With Ridge regression the conclusions are almost the same as in Lasso, as expected, meaning that the features with less importance in predicting our target are *Birth\_Year* and *Age*, while *Weight* and *BMI* can be considered as not important as well. With elastic net, we conclude that the features with less importance are *Age*, *Blood\_Pressure*, *High\_Cholesterol*, *Weight*, *Height,* and *Birth\_Year*. Figure has the plotted importance.

We also used two decision trees to calculate the feature importance: one using entropy and the other using Gini (see figure). If we define the threshold at 0.05, the variables we should keep are *Weight*, *Age*, *Blood\_Pressure*, *Smoking\_Habit*, *Exercise*, *Fruit\_Habit* and *Education*.

Joining all these insights from the different methods we concluded to keep a certain set of features and discard others. However, the results (see table) are just a guideline because we can try multiple combinations of features and see which one is the best for our models.

# Modelling

The main procedure is training the model by providing both features and a target for some observations, and then testing the model by only providing features and expecting it to predict the target. Therefore, we need to split the data into training and validation subsets. Since we already performed feature selection, we started by updating our dataframe to only contain the features that we selected. As we dropped *Education*, we do not need to fill in the missing values, and because we dropped *Birth\_Year* and *Age*, the outliers we defined before no longer apply.

As explained before, if we split the data using hold-out method (*train\_test\_split* function), we can only train a model with the portion set aside for training, which cannot fully represent our dataset. So, we used **cross-validation** (*K-Fold*, *Stratified K-Fold*, *Repeated K-Fold*, and *Repeated Stratified K-Fold*) and decided to use either *Stratified K-Fold* or *Repeated* *Stratified K-Fold*, as we wanted to make sure that the proportion of the different target labels was relatively the same in each subset created. In more complex models we could not use *Repeated* *Stratified K-Fold* as it was more time-consuming.

Then we defined 3 functions: the first one to evaluate the performance of a model, given a split method, an encoder schema and whether or not should we scale the data (the function returned the average f1-score for the training and validation dataset, the time taken to do the computations and the overfitting value); the second one to show the performances in a single dataframe; and the last one to find the best parameters to provide to a certain model by performing *GridSearchCV*.

We started by defining all the models used with the default parameters, to have an idea of how the models are going to perform on this dataset, and what are the most suitable ones in this case. Looking at the results of the **base models** (see table) we can say: generally, the most simple models are the ones that take less time, but they are not performing that well compared to others; we are quite overfitting in almost every model we have here since the difference between the training f1-score and the validation f1-score is quite high; comparing this neural network and the bagging classifier, we conclude that the performance is almost the same, but the bagging classifier works much faster than the neural network, so we should prefer it; also, we can say that since the KNN model has this kind of results with the default parameters, we can try to tune it and see if we can achieve better results with a really simple model. Lastly, our best models are *Bagging*, *Random Forest*, *Extra Trees* and *GradientBoosting*.

We performed Logistic Regression with different methods of split, and regarding probabilistic classifiers, we tried the basic Gaussian Naïve Bayes, Gaussian Naïve Bayes using only the metric features, and Categorical Naïve Bayes using only the non-metric features. As for the K-nearest neighbours (considering only the metric features) we found that the best performance of the algorithm was using 9, 11, or 14 neighbours, remembering that using an even number can be problematic as we can fall into a situation where the number of neighbours for each class is the same. We also tried to include the non-metric features by creating a model where the metric used was Jaccard similarity (see annexes). For neural networks we tried different combinations of parameters and for decision trees, we had a problem with overfitting, as we it was expected since we built fully grown-up trees. To overcome that, we tried to preprunning (changing parameters like ‘max\_depth’, ‘min\_samples\_split’, ‘max\_features’, etc.) and postpruning (using cost\_complexity\_pruning\_path) the trees we were building. Support vector machines was not one of the best base models, so we choose not to tune it. We tried to hyper-tune the Ridge classifier (see annexes), but the results were not the best.

For ensemble techniques, we tried Bagging, Random Forest, AdaBoost, GradientBoosting, Extra Trees (see annexes), and Stacking (combining in pairs the best models found before). These (except AdaBoost) were the models that performed better with the default parameter, so we hyper-tuned them all using the function we created. The full results can be found in table.

Preprocessing can have a significant impact on the performance of the models, and different preprocessing steps can be effective depending on the specific dataset and model being used. So, it is often useful to try a few different combinations and compare the results to see which combination of preprocessing steps works best. We tried different combinations of scaling methods for metric features and to remove or not the outliers with IQR method. The results can be found in table , and as we can see they are better when we do not remove the outliers. Regarding the scalers, the differences are not significant, as expected because scaled data is not needed for tree-based models, so we decided to keep our initial choice.

As we saw in the feature selection, the conclusion about whether we should discard or keep some features was dubious, therefore we tried our best models with just the features that we were sure to keep. The results can be found in table, and as we can see the results are slightly better when we use all features that were potentially selected, so we will keep them.

# Assessment

As found in the table, we can compare the different models that we tuned in the modelling phase to decide which one is the best. Five models have a f1-score of 1 in the training dataset, and around 0.98 in the validation dataset, so we should decide for the model that has better trade-off regarding overfitting time. We used the ROC curve and the AUC to compare the models, keeping in mind that these metrics are not sensitive to the cost of misclassification. So, as we can see in figure the model with the higher AUC is *GradientBoosting*, with parameters: *random\_state* = 5, *learning\_rate* = 0.1, *loss* = 'exponential', *max\_depth* = 20, *max\_features* = 'sqrt', *n\_estimators* = 100. The AUC in this case is 1, meaning that the probability that this model will assign a higher rank to a randomly selected positive instance than to a randomly selected negative instance is 1. We also plotted the precision-recall curve, and as we can see in figure , the best threshold to decide whether our model will consider an observation to have a target value of 1 or a target value of 0 is 0.176123 (usually the threshold used by the models is 0.5).

To submit our predictions to Kaggle, we must fit the model to our entire training dataset (as validation was already used to assess the performance). In the test dataset we must apply the same preprocessing steps we applied to the training dataset (creating new features, replacing the incoherences in *Region* variable, dropping the variables that were not used, scaling the metric features, and encoding the non-metric features). Then, we created the model with the best parameters and fit it to all the training data and made the predictions using the ‘predict’ method. The private f1-score in Kaggle was 1, but we expect a decrease in this value for the public score, since the distribution of the test data could be significantly different from the distribution of the training data, therefore the model may perform poorly on new test data even if it has learned the underlying patterns in the training data. We tried to minimize the overfitting considering the constrain that we had regarding the quantity of training data, by using cross-validation, regularization, and simple models and by trying to tune our parameters.

# Conclusion

In conclusion, we are aware the preprocessing and treatment of the features is a crucial step in obtaining these results, so we tried our best to treat the data, without removing a significant portion or without distorting the dataset. As we had a small train dataset, we had to deal with big variance and inconsistency in the obtained results, which was problematic when controlling the overfitting of the models. Regarding the context of the problem, some features could be better collected, like *Checkup* could have more relevant categories, and *Smoking\_Habit* could be a continuous variable.

After testing various models, with our best model (*GradientBoosting*), we predicted for the test sample that 115 patients have the Smith Parasite and the others 110 haven’t.

# References

Aznar, P. (17 de June de 2020). *QuantDare*. Obtido de What is the difference between Extra Trees and Random Forest?: https://quantdare.com/what-is-the-difference-between-extra-trees-and-random-forest/

Brems, M. (17 de April de 2017). *Towards Data Science*. Obtido de A One-Stop Shop for Principal Component Analysis: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

Brownlee, J. (12 de June de 2020). *Machine Learning Mastery*. Obtido de Ordinal and One-Hot Encodings for Categorical Data: https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/

Cook, A. (14 de December de 2021). *Kaggle*. Obtido de Scaling and Normalization: https://www.kaggle.com/code/alexisbcook/scaling-and-normalization

*Data Science StackExchange*. (9 de March de 2021). Obtido de How to understand ANOVA-F for feature selection in Python. Sklearn SelectKBest with f\_classif: https://datascience.stackexchange.com/questions/74465/how-to-understand-anova-f-for-feature-selection-in-python-sklearn-selectkbest-w

*Geeks for Geeks*. (15 de May de 2021). Obtido de Implementation of Lasso, Ridge and Elastic Net: https://www.geeksforgeeks.org/implementation-of-lasso-ridge-and-elastic-net/

*GitHub*. (26 de April de 2020). Obtido de Understanding Feature Mapping on Principal Components: https://github.com/anirb1nag/ML-Hobby-Projects/blob/master/Understanding%20Feature%20Mapping%20on%20Principal%20Components.ipynb

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining. Concepts and Techniques.* Elsevier.

Kapoor, H. (29 de May de 2020). *Kaggle*. Obtido de Random Forest vs Extra Trees: https://www.kaggle.com/code/hkapoor/random-forest-vs-extra-trees/notebook

Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics.* Massachusetts: The MIT Press.

Lee, W.-M. (18 de October de 2021). *Towards Data Science*. Obtido de Statistics in Python — Using ANOVA for Feature Selection: https://towardsdatascience.com/statistics-in-python-using-anova-for-feature-selection-b4dc876ef4f0

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2022). *Scikit Learn*. Obtido de sklearn.feature\_selection.mutual\_info\_classif: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.mutual\_info\_classif.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2022). *Scikit Learn*. Obtido de Ridge regression and classification: https://scikit-learn.org/stable/modules/linear\_model.html#ridge-regression-and-classification

*PyShark*. (s.d.). Obtido de Jaccard similarity and Jaccard distance in Python: https://pyshark.com/jaccard-similarity-and-jaccard-distance-in-python/

*Stack Overflow*. (24 de December de 2018). Obtido de What does sklearn "RidgeClassifier" do?: https://stackoverflow.com/questions/53911663/what-does-sklearn-ridgeclassifier-do

Swamynathan, M. (2017). *Mastering Machine Learning with Python in Six Steps.* Apress.

Thankachan, K. (6 de August de 2022). *Towards Data Science*. Obtido de What? When? How?: ExtraTrees Classifier: https://towardsdatascience.com/what-when-how-extratrees-classifier-c939f905851c

Verma, Y. (15 de August de 2021). *Analytics India Magazine*. Obtido de Hands-On Tutorial on ElasticNet Regression: https://analyticsindiamag.com/hands-on-tutorial-on-elasticnet-regression/

Verma, Y. (5 de April de 2022). *Analytics India Magazine*. Obtido de A hands-on guide to ridge regression for feature selection: https://analyticsindiamag.com/a-hands-on-guide-to-ridge-regression-for-feature-selection/

Zhu, A. (29 de June de 2021). *Towards Data Science*. Obtido de Select Features for Machine Learning Model with Mutual Information: https://towardsdatascience.com/select-features-for-machine-learning-model-with-mutual-information-534fe387d5c8

Zinda, Z. (4 de October de 2021). *phData*. Obtido de Data Science Stats Review: Pearson’s, Kendall’s, and Spearman’s Correlation for Feature Selection: https://www.phdata.io/blog/data-science-stats-review/

# 

# Annexes

Table 1 Features

|  |  |
| --- | --- |
| **Variable** | **Description** |
| *Disease* | Dependent variable, indicates if the user has the disease or not. It is only available in the train. |
| *PatientID* | Unique identifier of the patient |
| *Birth\_Year* | Year of the Birth of the patient |
| *Name* | Name of the patient |
| *Region* | Living Region of the patient |
| *Education* | Highest Education Level of the patient |
| *Height* | Patient’s height |
| *Weight* | Patient’s weight |
| *Checkup* | How long it has been since the patient last visited a doctor for a routine Checkup |
| *Diabetes* | If the patient or any directly related individuals have had diabetes |
| *High\_Cholesterol* | Patient’s Cholesterol value |
| *Blood\_Pressure* | Patient’s Blood Pressure in rest value |
| *Mental\_Health* | How many days in the past 30 days was the patient’s poor physical or mental health keeping them from doing usual activities, such as self-care, work, or recreation |
| *Physical\_Health* | How many days in the past 30 days was the patient’s physical health (physical illness and injury) not good to the point where it was difficult to walk |
| *Smoking\_Habit* | If the patient smokes more than 10 cigars daily |
| *Drinking\_Habit* | Patient’s behaviour concerning alcohol consumption |
| *Exercise* | If patient exercises (more than 30 minutes) 3 times per week or more |
| *Fruit\_Habit* | Portions of fruit the patient consumes per day |
| *Water\_Habit* | Patient’s water consumption per day |

Table 2 Metric features - descriptive statistics



Table 3 Categorical features - descriptive statistics



Table 4 Categorical and ordinal features

|  |  |
| --- | --- |
| **Categorical** | **Ordinal** |
| *Region* | *Education* |
| *Smoking\_Habit* | *Drinking\_Habit* |
| *Exercise* | *Fruit\_Habit* |
| *Diabetes* | *Checkup* |
| *Gender* | *Water\_Habit* |

Chart, bar chart, histogram

Description automatically generated

Figure 5 Numerical variables - histograms

Diagram

Description automatically generated with low confidence

Figure 6 Numerical variables – box plots

Diagram

Description automatically generated

Figure 7 Numerical variables – pairwise relationships

Chart, waterfall chart

Description automatically generated

Figure 8 Non-metric variables - count plots



Figure Numerical variables – pairwise relationships by disease

A picture containing text, crossword puzzle

Description automatically generated

Figure Numerical variables – box plots by disease

A picture containing text, crossword puzzle

Description automatically generated

Figure Numerical variables – histograms by disease

Chart

Description automatically generated

Figure Non-metric variables – absolute frequencies by disease

Table New features

|  |  |
| --- | --- |
| **Variable** | **Formula** |
| Age: gives the age of the patient based on the variable *Birth\_Year* and the current year | df['Age'] = date.today().year - df['Birth\_Year'] |
| Gender: gives the gender of the patient (assuming binary gender), based on the presence of Mrs. or Mr. on the variable *Name*. | df['Gender'] = np.where(df['Name'].str.contains('Mr\.'), 'Male', 'Female') |
| BMI: (Body Mass Index) given by: | df['BMI'] = df['Weight'] / ((df['Height'] / 100) \*\* 2) |

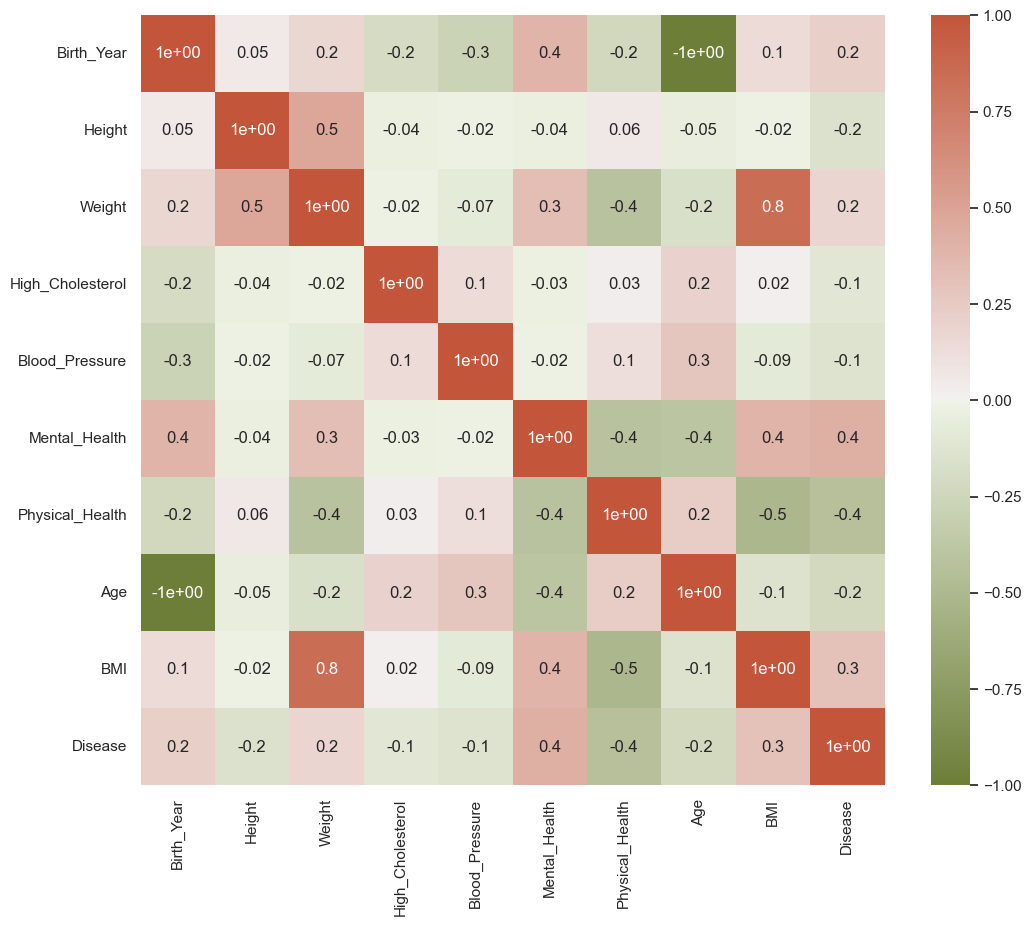


Figure Spearman correlation matrix

Table - ANOVA's results



Table Kendall's results



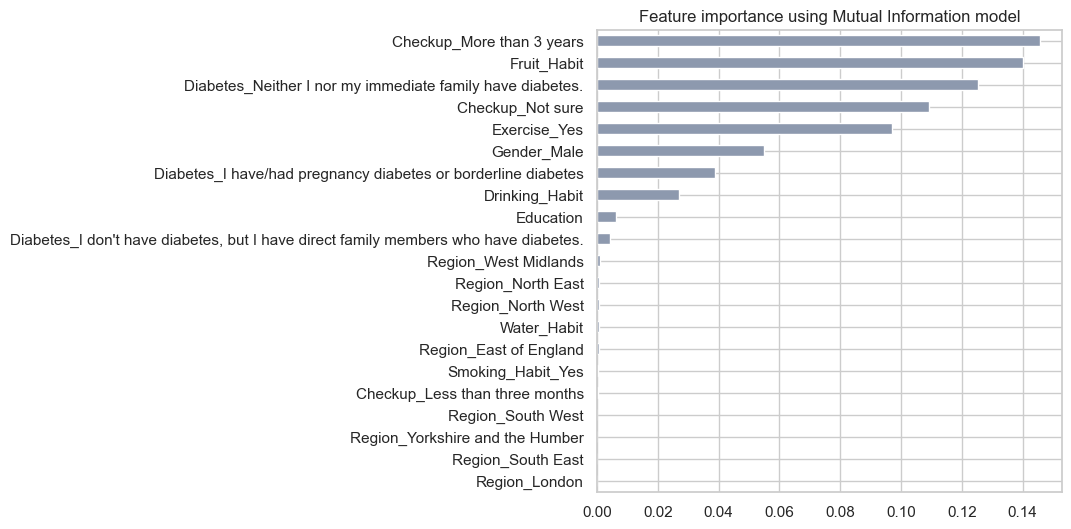


Figure Mutual information importance

**Explanation for the plots below**: we created two plots for each feature, where the first plot represents the frequency of each value of the target in the different categories, and the second plot checks the proportion of the presence of each label in each category.

* People that do not visit the doctor for more than 3 years have higher probability to have the disease than the others, so maybe *Checkup* is an important variable to predict the target.
* We do not have a specific region that has significantly larger proportion of patients with the disease, so maybe *Region* is not an important variable to predict the target.
* The same happens with *Smoking\_Habit.*
* For the alcohol, we can say that the group which consumes alcohol every day has a slightly bigger proportion of patients with the disease than the others, so maybe *Drinking\_Habit* can help to predict the target.
* People that don't exercise at least 3 times per week have higher probability to have the disease than the ones that do so, therefore maybe *Exercise* is an important variable to predict the target.
* People that eat lots of fruit per day and people that do not eat fruit at all are the two groups where the presence of the disease is higher (which is strange), so we can extrapolate that maybe *Fruit\_Habit* can predict the target.
* We do not have a specific type of water consuming that has significantly larger proportion of patients with the disease, so maybe *Water\_Habit* is not an important variable to predict the target.
* The same happens with *Education*.
* Finally, for the variable *Diabetes*, we cannot be sure, since we have a specific group (the group that neither them nor their family have diabetes) that clearly have a greater tendency for not having the disease, but the other groups are really tight in terms of performance.

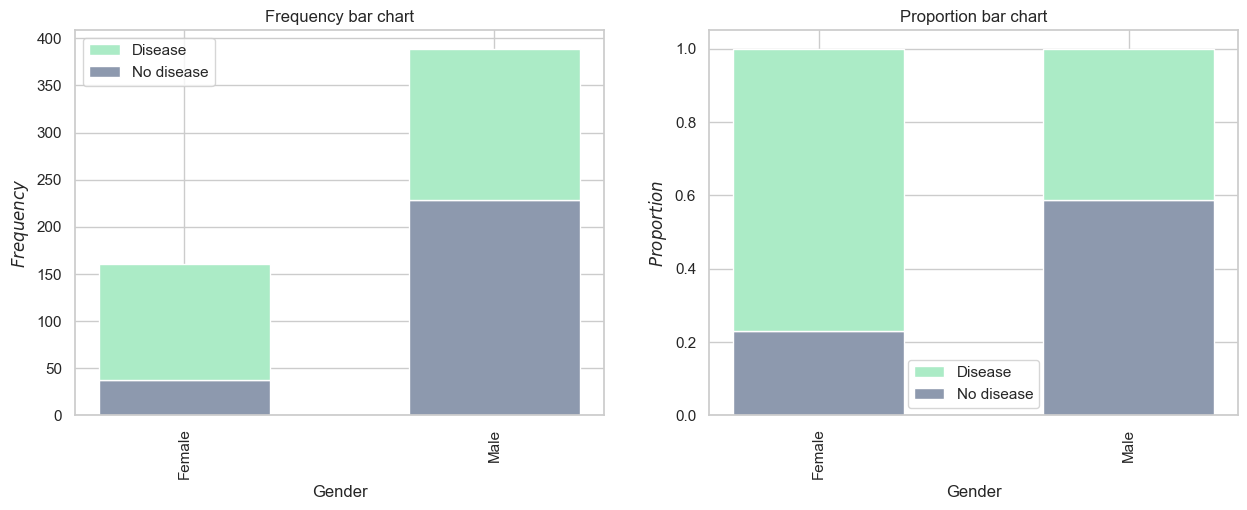


Figure Weight of the dependent variable in Gender

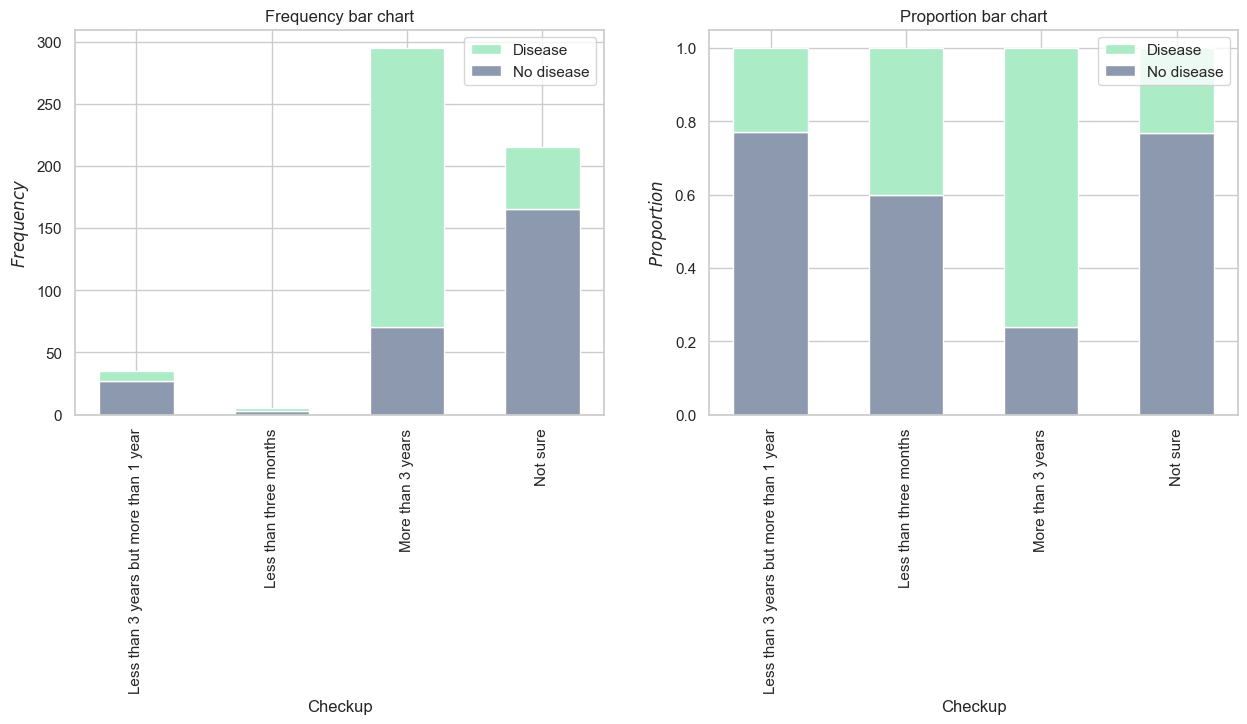


Figure Weight of the dependent variable in Checkup

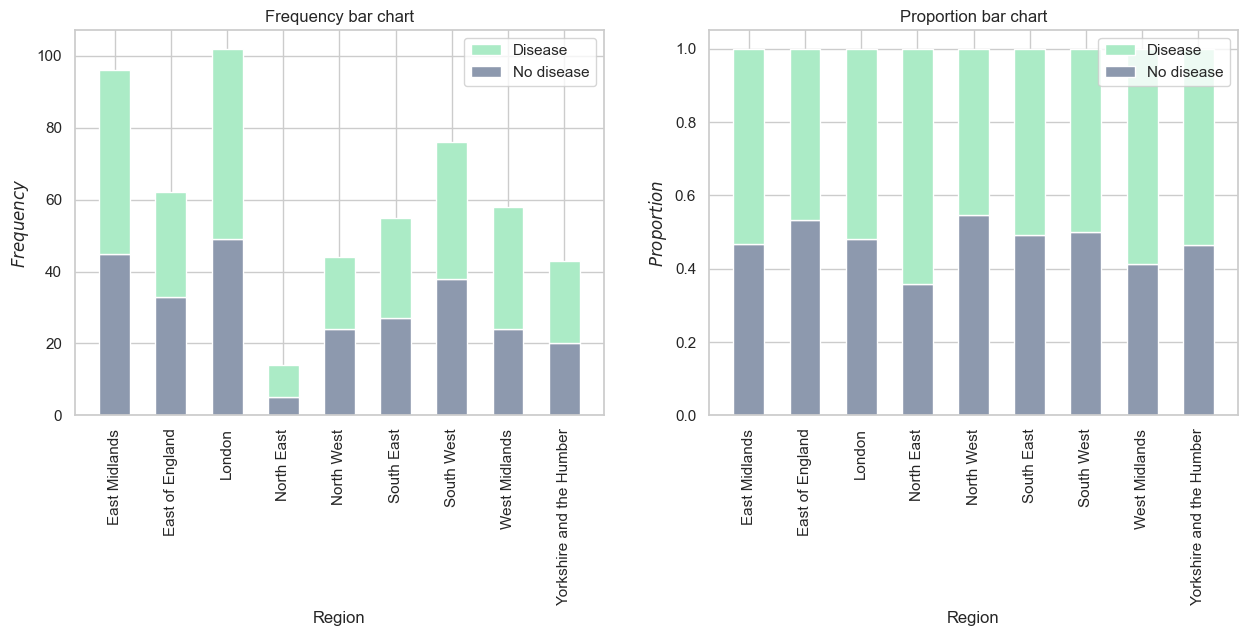


Figure Weight of the dependent variable in Region

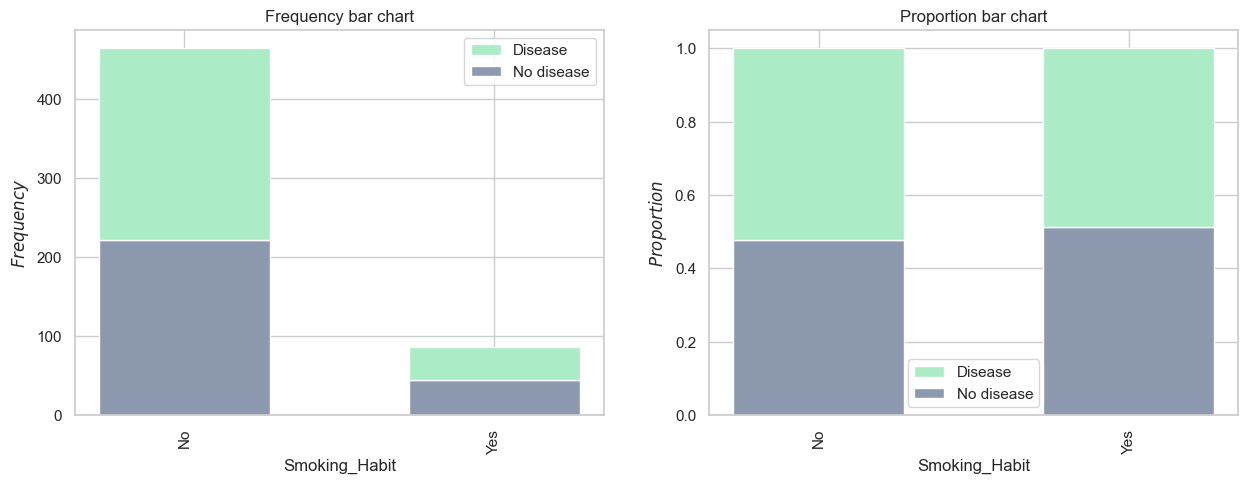


Figure Weight of the dependent variable in Smoking\_Habit

*Chart, bar chart

Description automatically generated*

Figure Weight of the dependent variable in Drinking\_Habit

Chart, box and whisker chart

Description automatically generated

Figure Weight of the dependent variable in Exercise

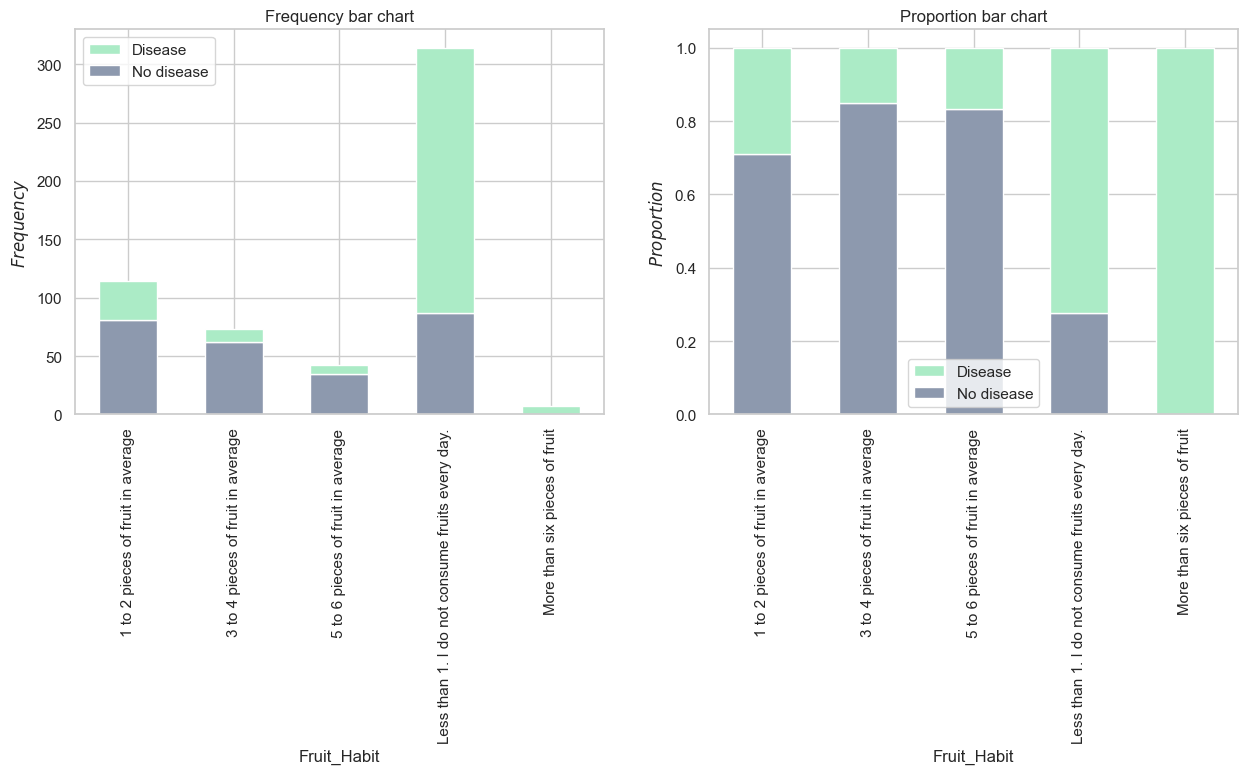


Figure Weight of the dependent variable in Fruit\_Habit

Chart, bar chart, waterfall chart

Description automatically generated

Figure Weight of the dependent variable in Water\_Habit

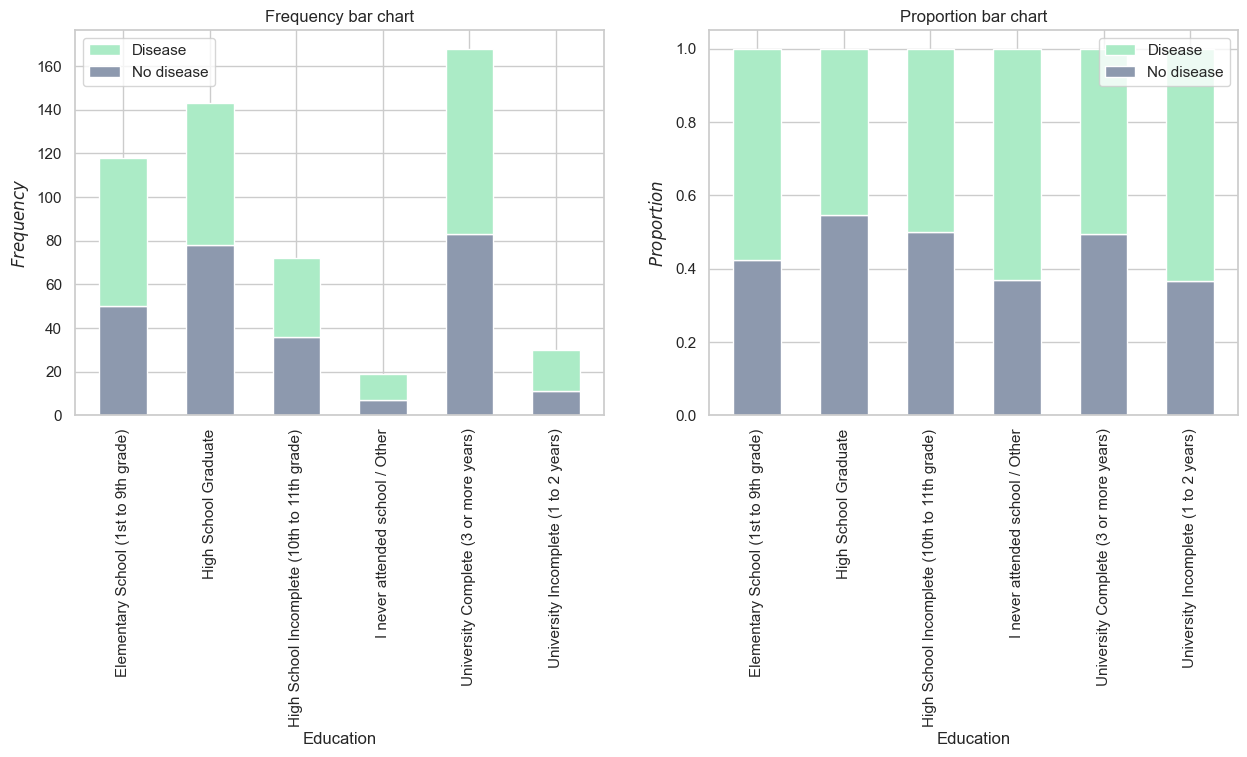


Figure Weight of the dependent variable in Education

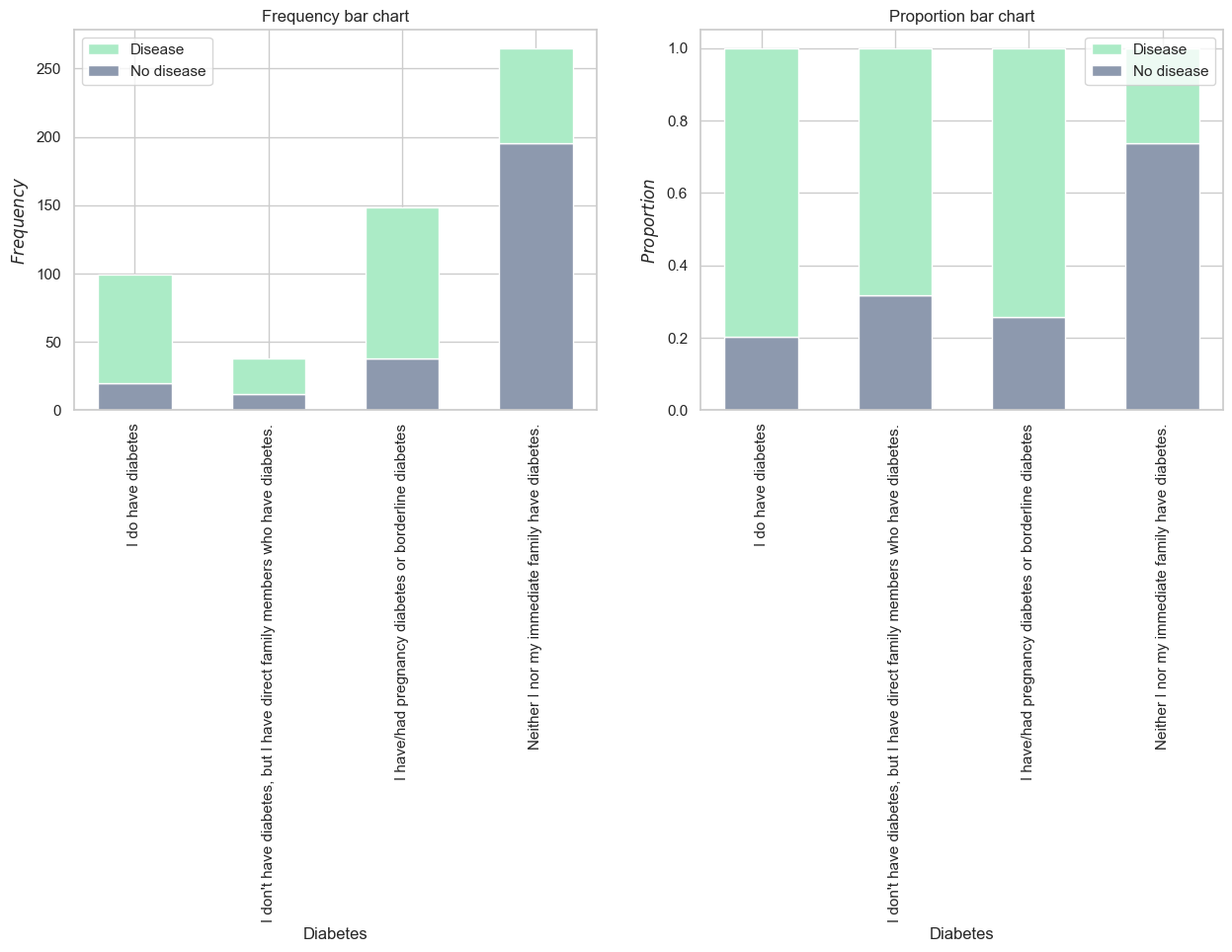


Figure Weight of the dependent variable in Diabetes

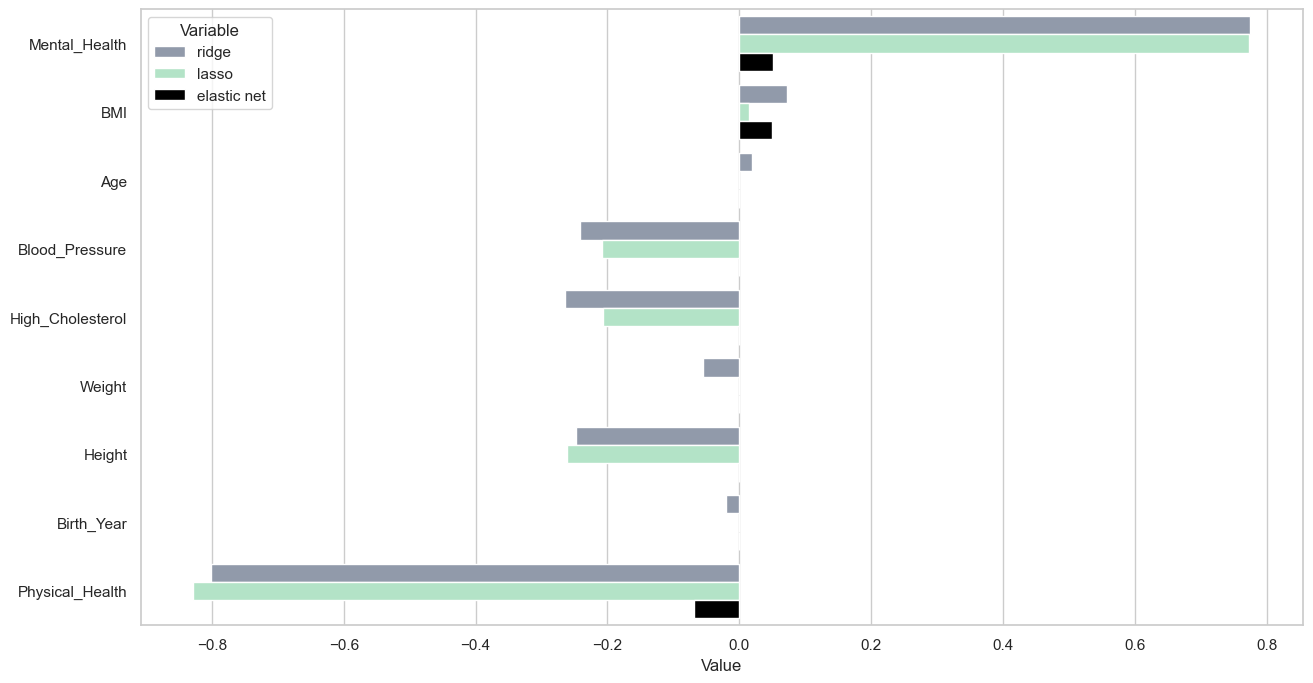


Figure Feature importance using Lasso, Ridge and Elastic Net

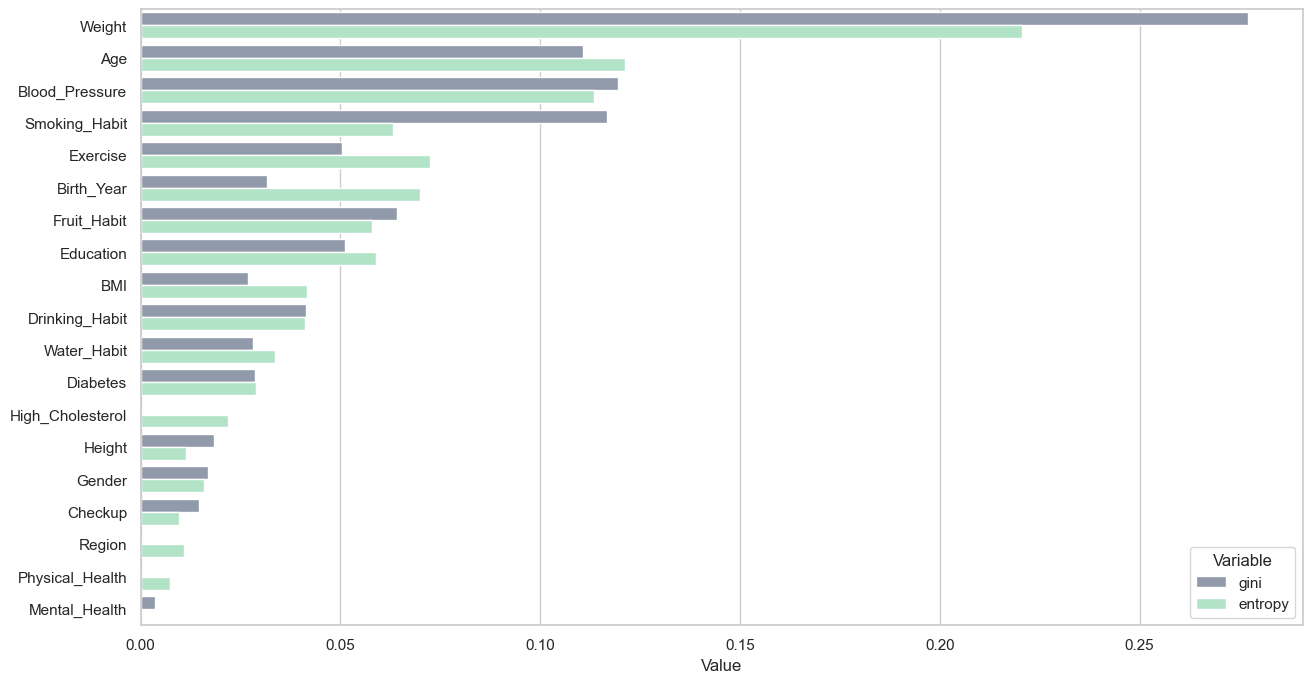


Figure Feature importance using Decision Trees

Table Feature selection conclusions for numerical data

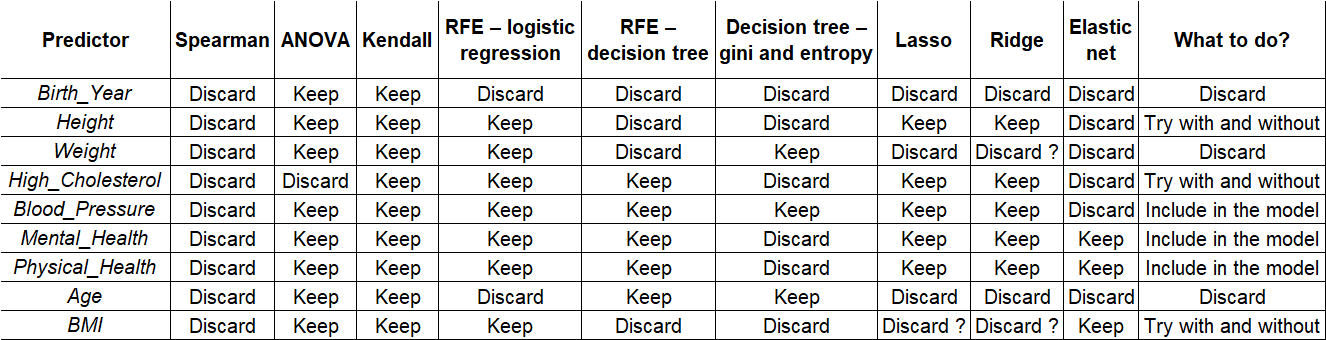


Table Feature selection conclusions for non-numerical data

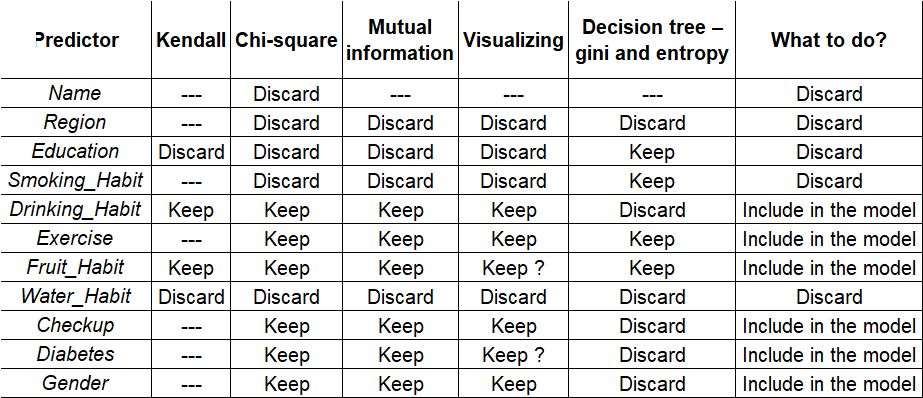


Table Results of the base models

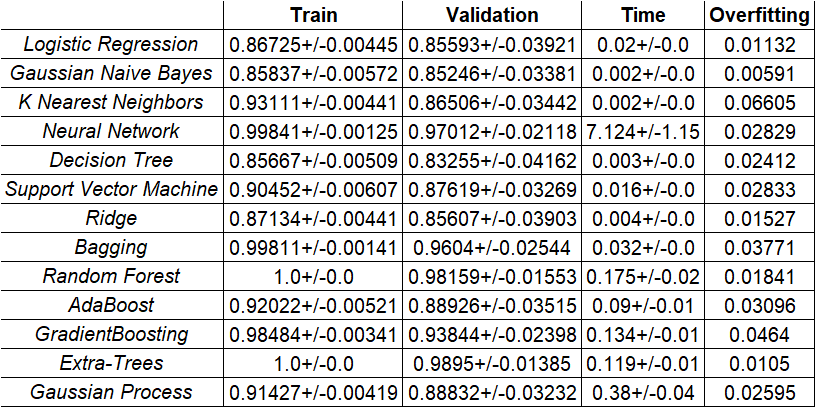


Table Results of the best models



Table Comparing the best models using all the features selected and only a part of them

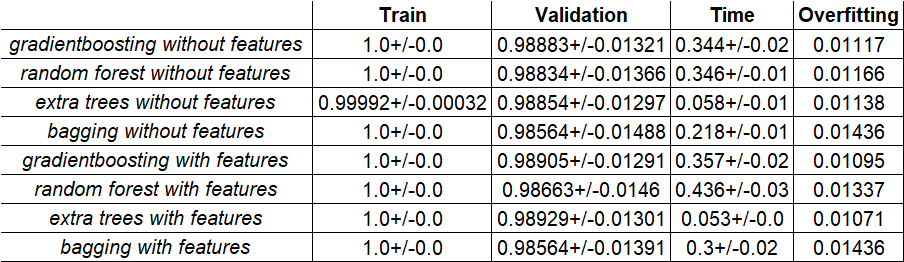


Table Results with different preprocessing combinations



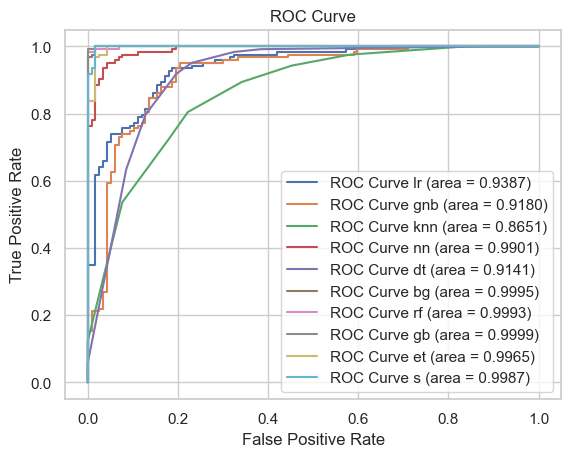


Figure ROC curve and AOC

Chart

Description automatically generated

Figure Precision-recall curve

## Self-study – data exploration

* In **Principal Component Analysis** we search for k n-dimensional orthogonal vectors that can best be used to represent the data. So, the original data is projected onto a smaller space, and then we can reduce the dimensionality of our data (Han, Kamber, & Pei, 2012). “PCA finds the directions of maximum variance in high-dimensional data such that most of the information is retained and projects it onto a smaller dimensional subspace” (Swamynathan, 2017). We applied PCA to the metric features in our dataset and we plotted the eigenvalues (the importance of its corresponding eigenvector) and the proportion of the variance explained (by each PC and the cumulative) and decided to use 4 principal components because of the elbow method.

Chart, line chart

Description automatically generated

Figure Elbow method for PCA

We can also see the correlation between the metric features and each principal component that we created. Looking at this dataframe we see that *Principal Component 1* behaves almost the same as *Height* but in an inverse way, *Principal Component 2* behaves almost the same as *Birth\_Year* and *Age* (so it nearly explains the age of the patients), *Principal Component 0* is the most informative component, as it is highly correlated with 4 of our metric features. We can also see that *High\_Cholesterol* is not really represented by a principal component.

Table Correlation between the metric features and the principal components



Here, we can visualize in a 2-dimensional projection of the multi-dimensional dataset that we have using the principal component analysis technique (GitHub, 2020), and then we visualize the contribution of each feature to the two principal components (visualization of the red and green dataframe below). "By projecting our data into a smaller space, we’re reducing the dimensionality of our feature space… but because we’ve transformed our data in these different “directions,” we’ve made sure to keep all original variables in our model" (Brems, 2017).

Chart, scatter chart

Description automatically generated

Figure Feature mapping with principal components

As we can see in the plot above, we have the whole dataset projected into the two first principal components, and we can visualize the contribution of each individual feature along these two principal components. The length of each feature vector represents how sensitive the principal components are towards that feature, while the direction represents which component is getting more affected (positively or negatively) by that feature. These results emphasize the previous ones presented on the red and green dataframe. We can see that *Height* is negatively affecting the *Principal Component 1* (vector is pointing below to the negative part of y-axis) and *Physical\_Health* is affecting positively the *Principal Component 0* (pointing right, to the positive part of x-axis), while *BMI*, *Mental\_Health* and *Weight* are pointing left, to the negative side of x-axis, meaning that they are negatively affecting the *Principal Component 0*. But, as we can see in the representation of the observations, we cannot distinguish very well the two target values using only the Principal Components (maybe because 2 PC only explain nearly 60% of the variance). So, the Principal Component approach may not be the best one.

## Self-study – feature selection

* **ANOVA** is a method that checks if the mean of two or more groups are significantly different from each other (Lee, 2021). The null-hypothesis is: the means of all groups are equal; and the alternative-hypothesis is: at least one mean of one group is different. So, if the mean of a metric feature regarding one category of the target is significantly different from the mean of the same metric feature regarding the other target category, we conclude that the feature has impact on the target, so we should keep it for the model training. And the other way around. We want to use features that have the highest possible F-score and the lowest possible p-value (if the p-value is lower than the statistical significance, in this case we will consider 5%, we reject the null-hypothesis, meaning that we have statistical evidence that at least one mean of one group is different, which is good for the predictive capacity of our feature) (Data Science StackExchange, 2021).
* The strength of the association between two variables is known as the correlation test, and we can measure non-parametric correlation (non-linear dependences) with **Kendall** (tau), measuring how likely it is for two variables to move in the same direction, but not necessarily at a constant rate. The Kendall's rank correlation coefficient is the test statistic when we want to test if two variables are statistically dependent. This test is non-parametric, so we do not need to depend on the distributions of the variables. Under the null-hypothesis, we have independence of X and Y, so the distribution of tau has mean of 0. However, as this test is based on the ranked values of each variable, it will work with numerical features, but it can also be used with ordinal features (Zinda, 2021).
* **Mutual information** measures the mutual independency of two random variables (Han, Kamber, & Pei, 2012) or the entropy drops subject to the target (Zhu, 2021). The mutual information score ranges between 0 and infinite, and the higher the value of mutual information, the better, because it means that the relationship between that certain feature and the target is close. Mutual information is “equal to zero if and only if two random variables are independent, and higher values mean higher dependency” (Pedregosa, et al., Scikit Learn, 2022). The problem with mutual information is that mutual information is not a metric and is not normalized, so it is difficult to compare it in different datasets.
* **Ridge regression** is a method to “estimate the coefficient of multiple regression models” (Verma, Analytics India Magazine, 2022). Both Lasso and Ridge regression work by penalizing the magnitude of coefficients of features along with minimizing the error between predictions and actual values or records, the biggest difference is that Lasso can reduce the coefficient of a feature to zero (eliminating it), while Ridge cannot (Ridge penalty is “equivalent to square of the magnitude of coefficients” (Brownlee, 2020)). So, we find that Ridge regression performs better when the data consists of features which are sure to be more relevant and useful.
* “Ridge regression added a term in ordinary least square error function that regularizes the value of coefficients of variables” (L2) (Geeks for Geeks, 2021), and “Lasso regression is similar to Ridge regression except we add mean absolute value of coefficients in place of mean square value” (L1) (Geeks for Geeks, 2021). The **elastic net** is a regularization method that combines the L1 (this penalty encourages sparsity) and L2 (this penalty encourages the selected features to be correlated) penalties of the Lasso and Ridge methods. Regularization is a technique to reduce the errors in the validation and testing sets by fitting a function (the penalties) appropriately in the training dataset (Verma, Analytics India Magazine, 2021).

## Self-study – models

* **Jaccard similarity**,also known as the Jaccard coefficient, is a measure of the similarity between two sets (PyShark, s.d.). Jaccard similarity is a useful measure of similarity for sets because it considers the presence and absence of elements in the sets, rather than just the count of elements. In the KNN algorithm, similarity is typically measured using the Euclidean distance, for example. However, Jaccard similarity can be used when the data being analysed consists of sets of binary values (as it is in the case of non-metric features encoded with one-hot encoding), because it is a metric specifically designed to measure the similarity between sets of binary values.
* **Ridge classifier** algorithm starts by converting the two classes in the target to {-1, 1}, and then treats the problem as a regression problem, minimizing a penalized residual sum of squares (Pedregosa, et al., Scikit Learn, 2022). If the regressor's prediction is positive, then the label of the observation will be 1, if the regressor's prediction is negative, the label of the observation will be -1. Ridge classifier is similar to logistic regression, but it is faster and can be very versatile, as it allows different numerical solvers. Usually, the main reason for overfitting is the model complexity, and regularization can control the model complexity, because it penalizes higher terms in the model, meaning that when a regularization term is added, the model will minimize not only the loss function but also the complexity. So, Ridge classifier introduces the L2 regularization (Stack Overflow, 2018). Instead of minimizing the cost function of the logistic regression, Ridge classifier penalizes the sum of the squared values of the weights, so if the weights take large values the optimization function is penalized.

, where is in this case:

* **Extra trees classifier** (Extremely Randomized Trees)is an ensemble learning method that constructs a multitude of decision trees (combination of multiple decision trees) during training and the output is the class that corresponds to the model of the classes (majority vote). Extra trees and random forests have a lot in common, as both of them are composed of numerous decision trees, the final decision is obtained taking into account the prediction of every tree, and when selecting the partition of each node, both of them randomly choose a subset of features (Thankachan, 2022). However, there are two major differences between these two algorithms:
  + Random forest uses subsamples of the data (bootstrap samples) to train the model, making sure that the decision trees are sufficiently different. Extra trees uses the entire dataset (we can change the parameter bootstrap, but by default, the sklearn implementation uses the entire input sample).
  + Extra trees randomly selects the value to split nodes, while random forest chooses the optimum split. But, once the split points are selected, both methods choose the best one between all the subsets of features (Aznar, 2020).

Knowing these differences, extra trees algorithm is faster since it randomly chooses the split point instead of calculating the optimal one. The decision trees in extra trees are usually bigger having more levels and nodes (Kapoor, 2020).